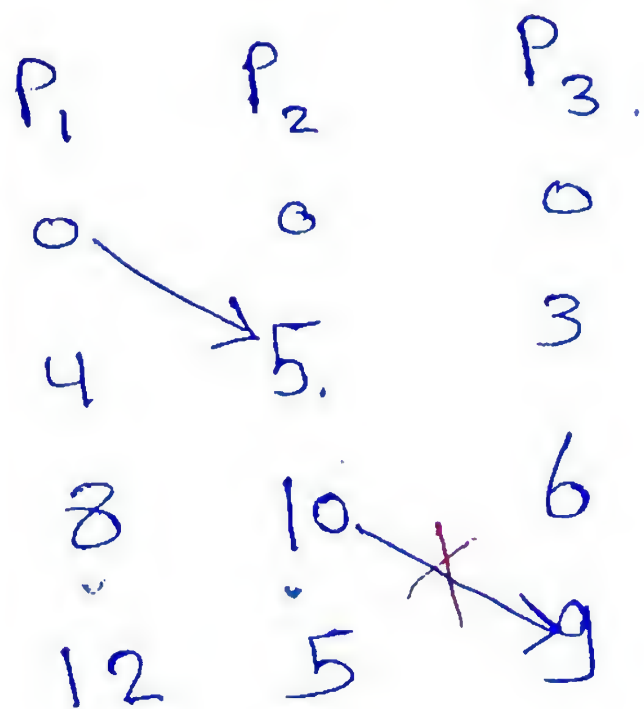


slide 4 Synchronization in Distributed system

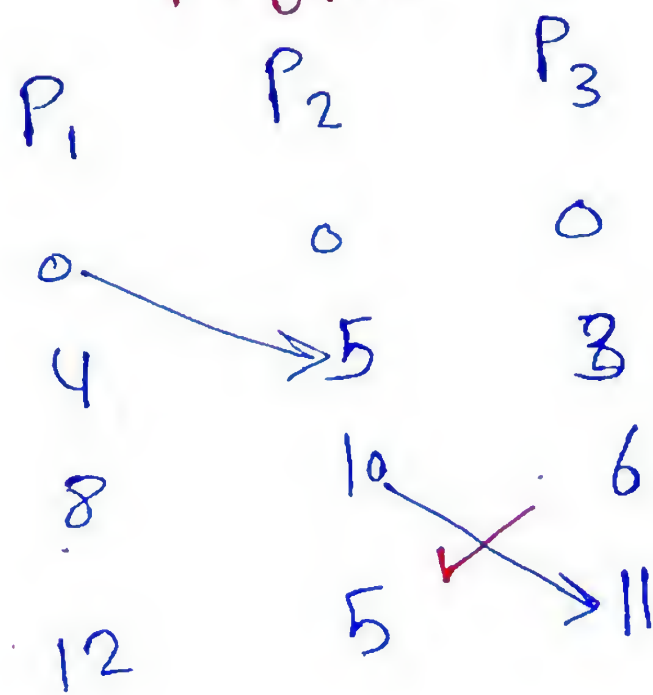
LamPort Logic clock:

When a machine send Message at t_0 to another machine at t_1 , so t_1 must be greater than t_0 Logically.

wrong



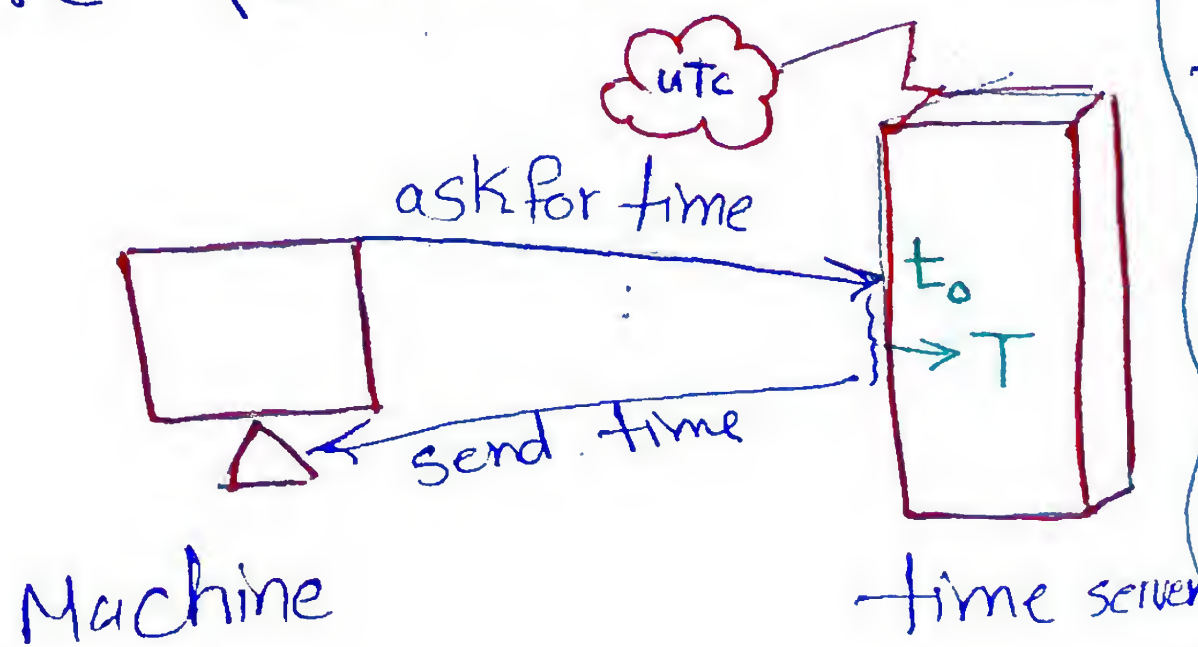
right



- two ways to maintain synchronization :

Cristian's Algorithm
(using UTC)

You have a time server
each machine ask him
for time and he references
UTC (universal coordinated time).



Assume :

$$RTT = \text{ask time} + \text{send time}$$

&

$$\text{ask time} = \text{send time}$$

$$\text{So Send time} = RTT/2$$

t_0 : request arrive time

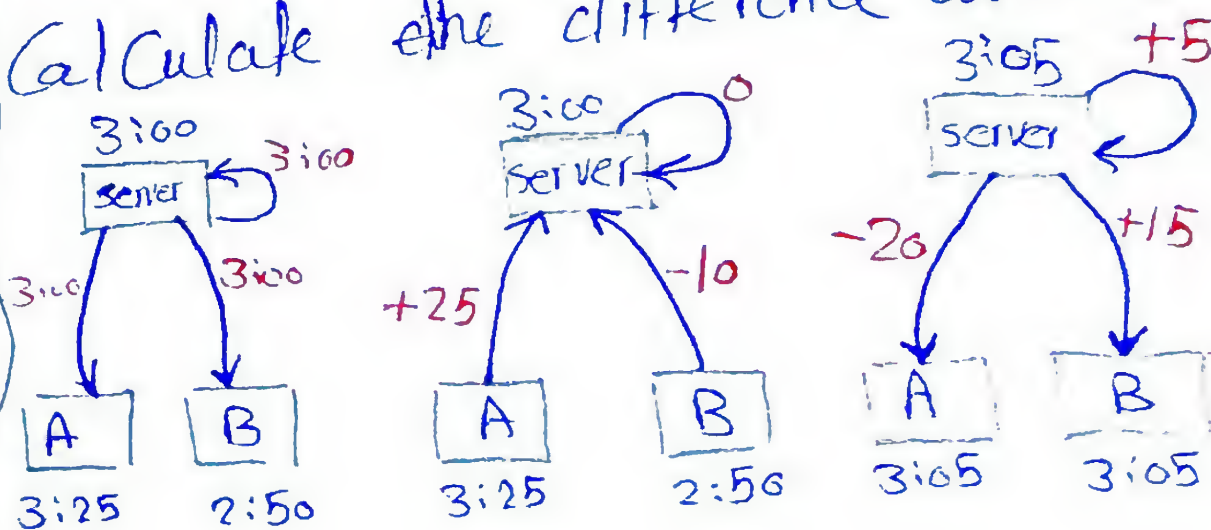
$$\text{given time to Machine} = t_0 + T + \frac{RTT}{2}$$

Berkeley unix
(without UTC)

1- centralized :

time server send its time
to all machines and they
respond by difference between
them then he sends corrected
time to each one after

Calculate the difference average



2- DeCentralized : all Machines

broadcast its time and calculate
the average and correct only its
own time

two APPROACHES to Maintain mutual exclusion of shared memory:

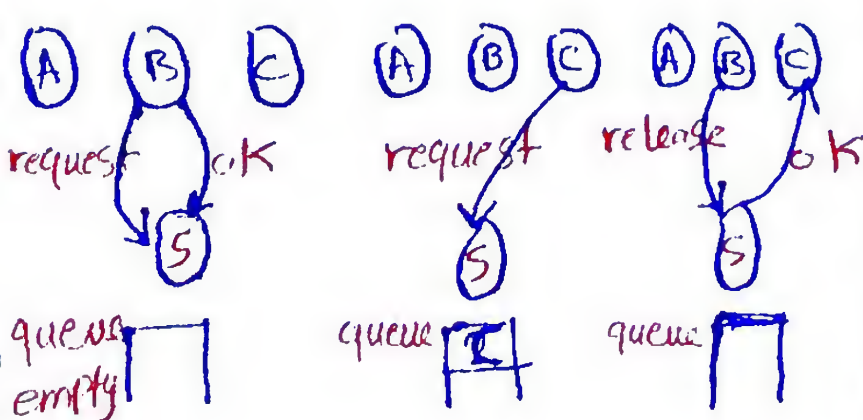
[1] Permission

[2] Token

Permission

Centralized

one Machine is elected as a Coordinator and any Machine want to access any resource, asking him and he reply with ok if the resource is free and No reply and Make a machine in the resource queue if it's used by another one, until it's released, then give him Permission.



advantage: • simple to implement

disadvantages: • single Point of failure
• bad Performance

Decentralized

each resource is replicated n times and with one Coordinator per replica.

when a machine want to access some resources search by resource name in hash table when machines have it and to have Permission to Access it must get Permission from ~~machine~~ resource Coordinator $> n/2$

advantage: No single Point of failure.

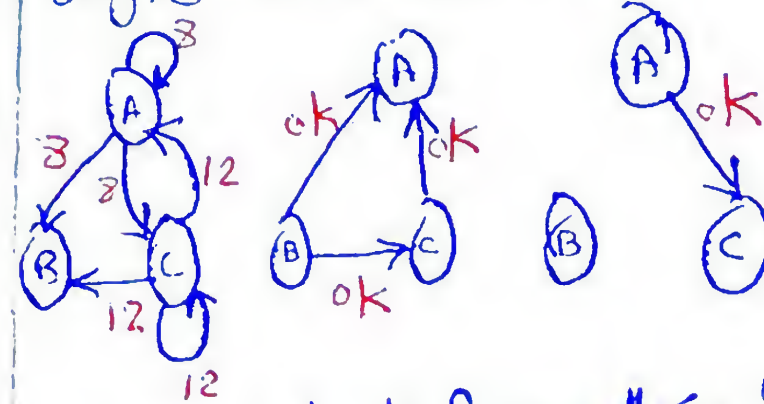
disadvantage: lots of Messages.

Distributed

resource requestor send Messages to all Machines with its Logic time and which resource he want.

if it's busy, he will get into queue.

if two Machine ask at the same time, the least Logic time will win.



if you get OK from all so you win.

advantage: • No centralization
• fewer Messages

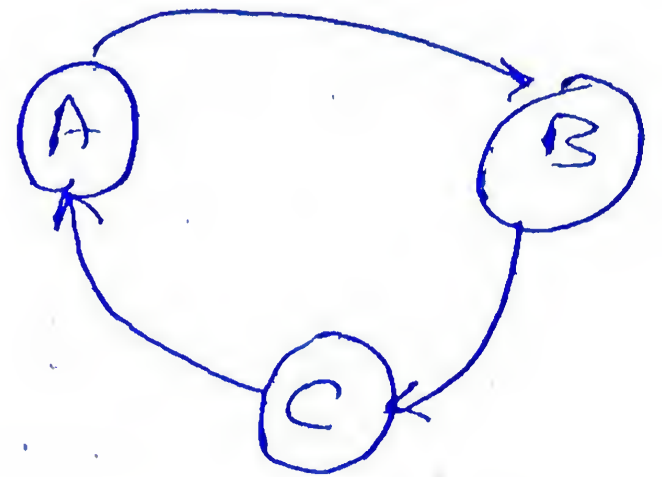
disadvantage: n Point of failure as you have get OK Message from all.

[2] Token System

organize the processes (Machines) in a ring and pass a token through the ring, the one has the token can access any resources if it ~~is~~ doesn't need it pass it to the successor

advantage:

- fairness (no starvation)



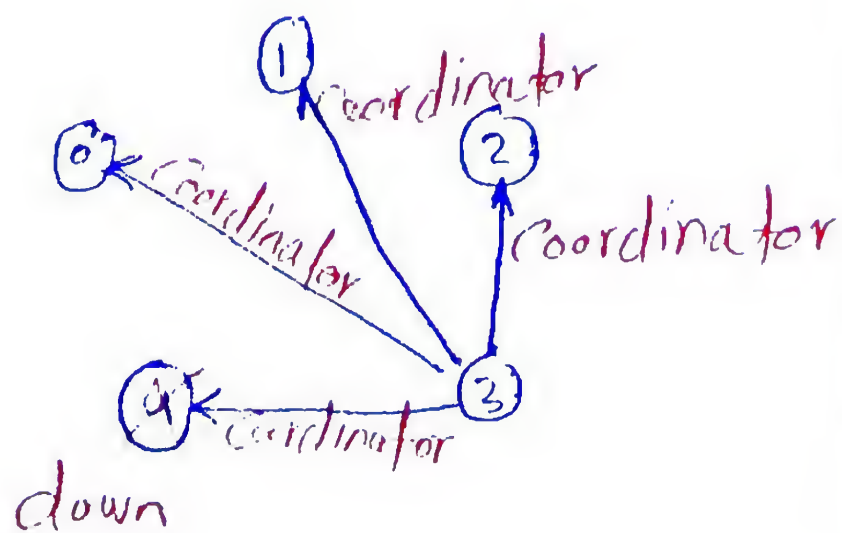
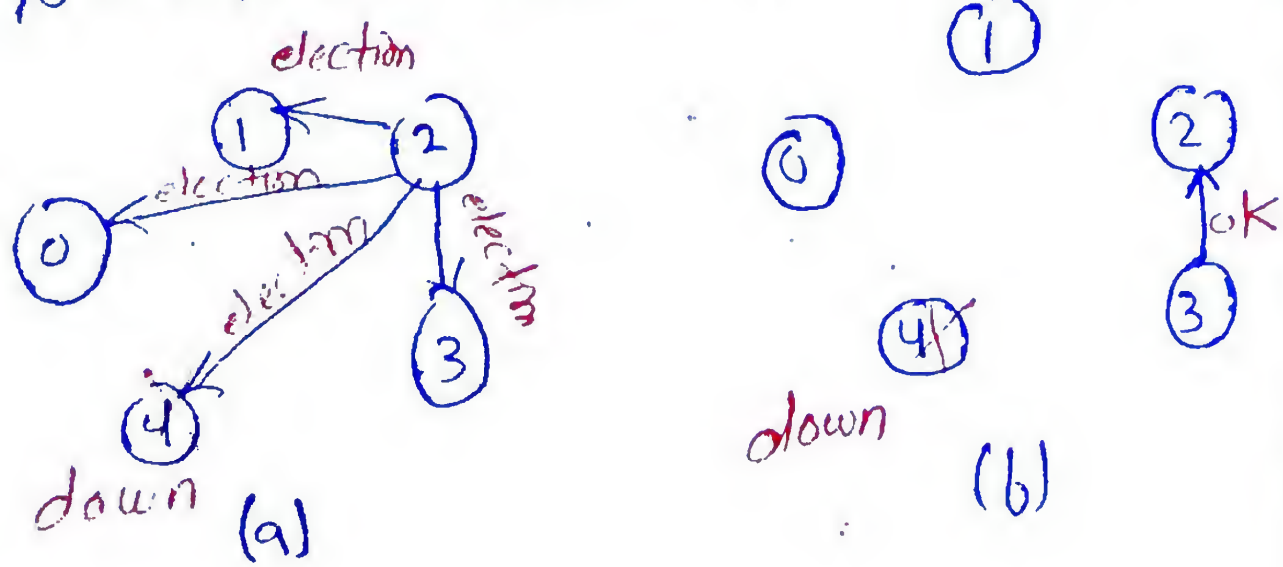
disadvantage:

- crash of a process holding token
- if token is lost, it's ~~then~~ be regenerated

Election algorithms for Coordinator:

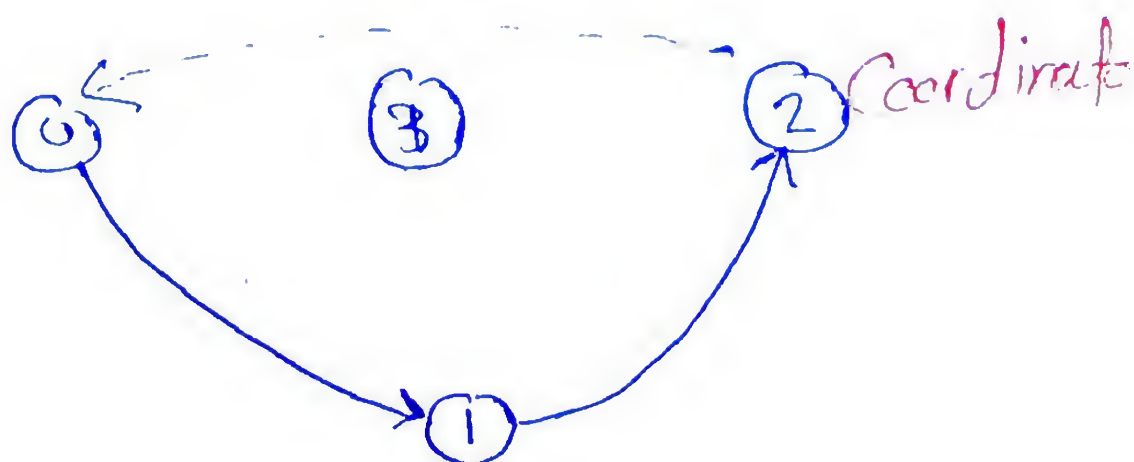
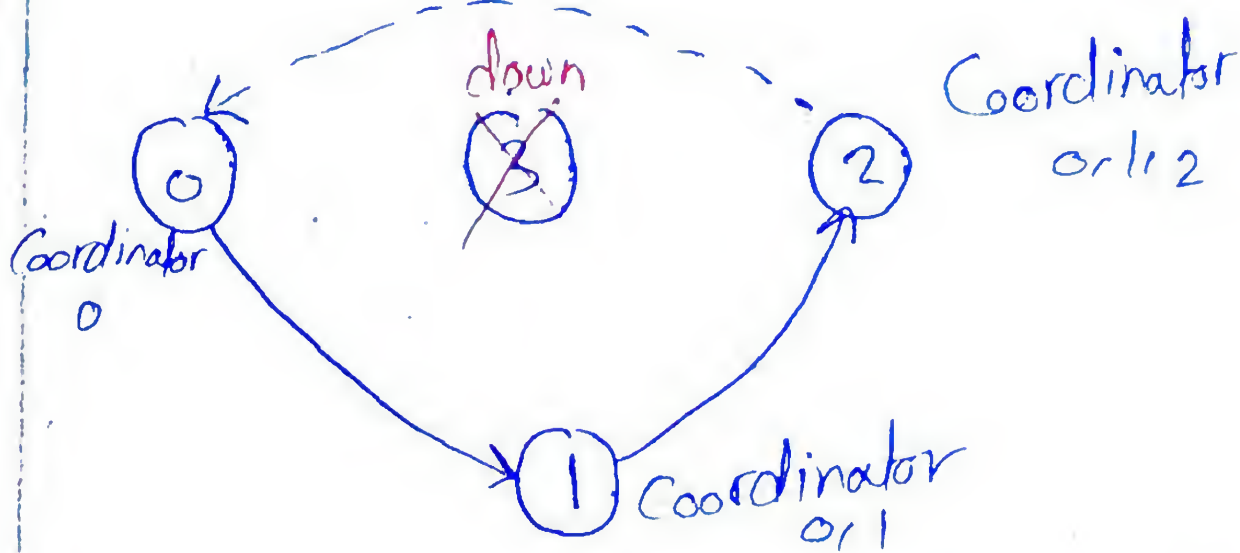
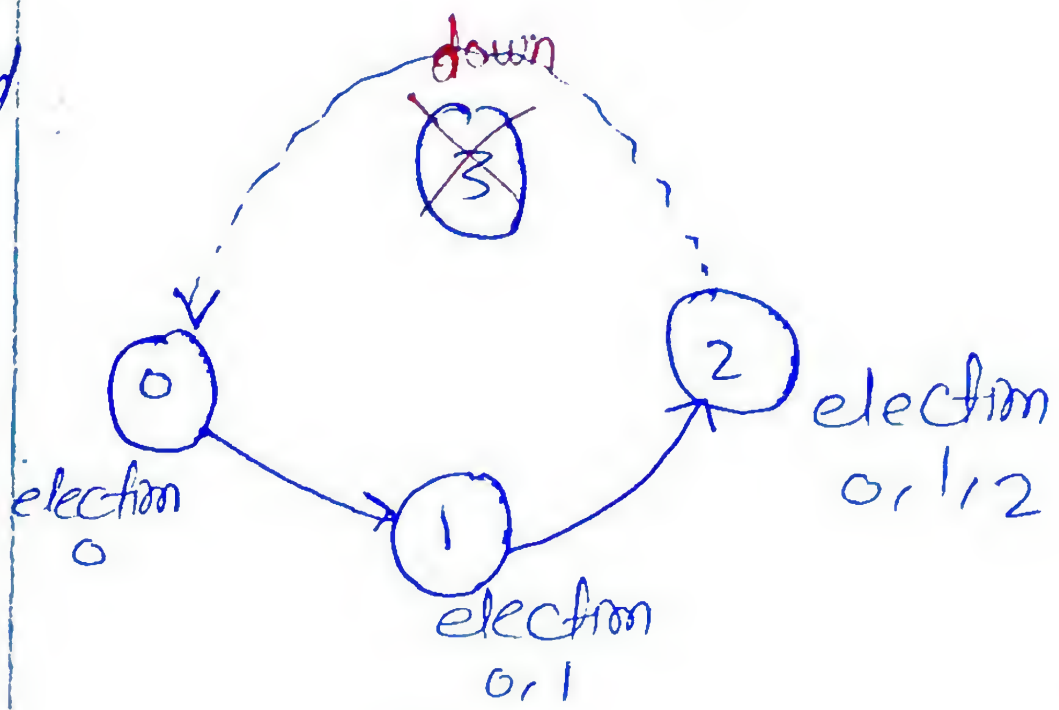
① Bully Algorithm

each process has a number, each one would like to be coordinator send message to upper-numbered processes and if any of them inside the election he send ok-message to it to tell it, he's alive and stop him until the biggest number when



② Ring Algorithm

each process also has a number, and each one would like to be coordinator send message and one with highest number, win the election.



Transaction: one or more operations happens
completely ^(commit) or not at all ^(rollback) (ex:

transfer money between banks a (counts)

types of transactions:

1. Flat transaction (processing on one database).
2. Nest transaction: (processing on two different DB).
airline DB hotel DB
3. Distributed transaction (processing on the same DB but separated physically)

Methods of implementing transaction:

[1] private workspace

You make changes on a copy of real data in your workspace after complete it, reflect them on real data. and if it's not completed, No effect on real data

disadvantage: cost of copy files to workspace.

[2] writeahead Log

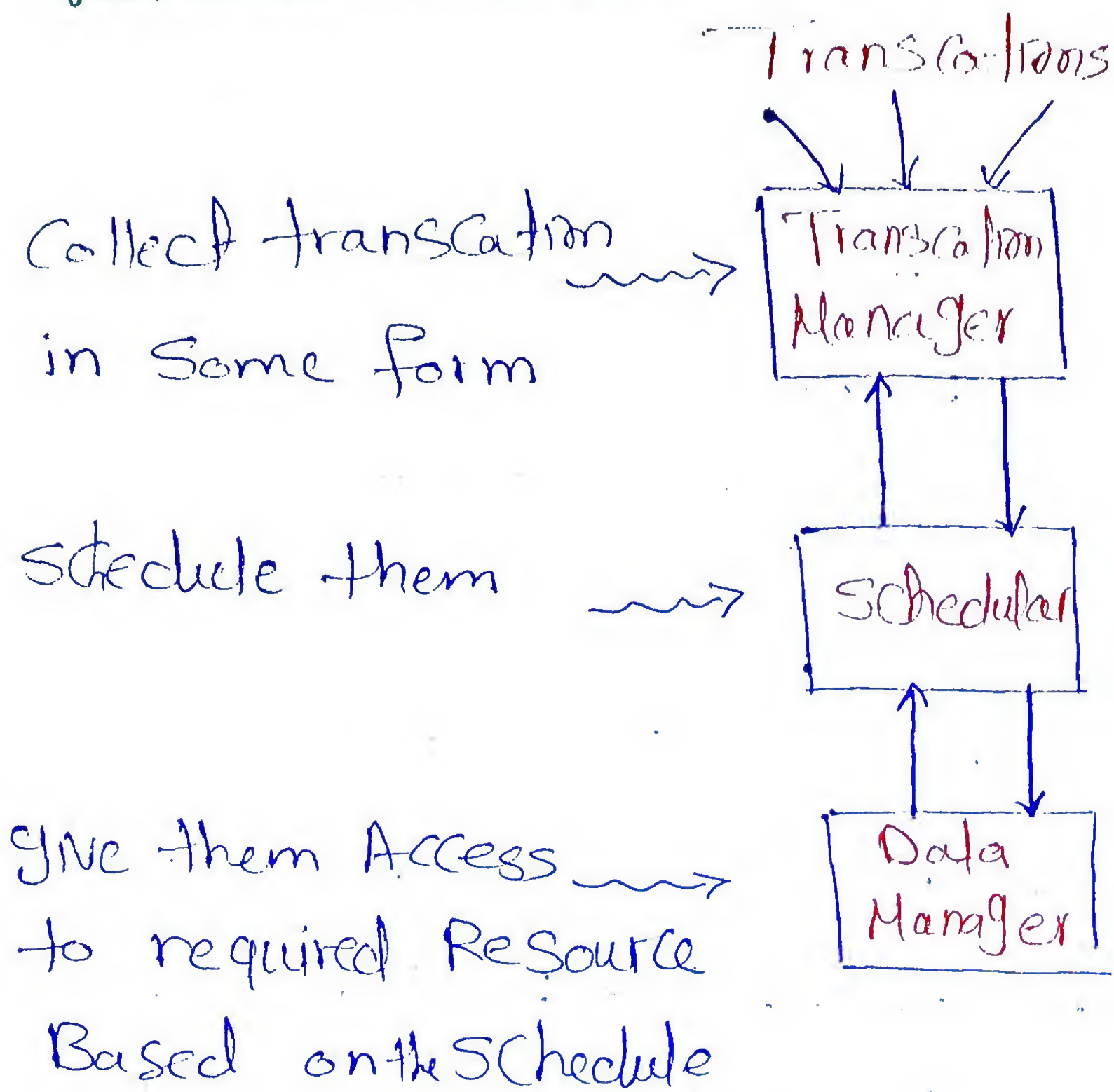
You make changes on the real data but you log its previous value before change, if it's aborted (not complete transaction) get back old value.

Two Phase Commit Protocol: most widely Protocol used for implement transaction in Distributed System.

Phase 1 :- The Coordinator send ~~the~~ Prepare message to all Machine to check if they are ready to implement transaction, if he get ready Messages from all of

Phase 2 :- them, he send Commit order to them after they finish, they Notifie him. If he doesn't get ready message from all, the transaction is aborted (Cancelled)

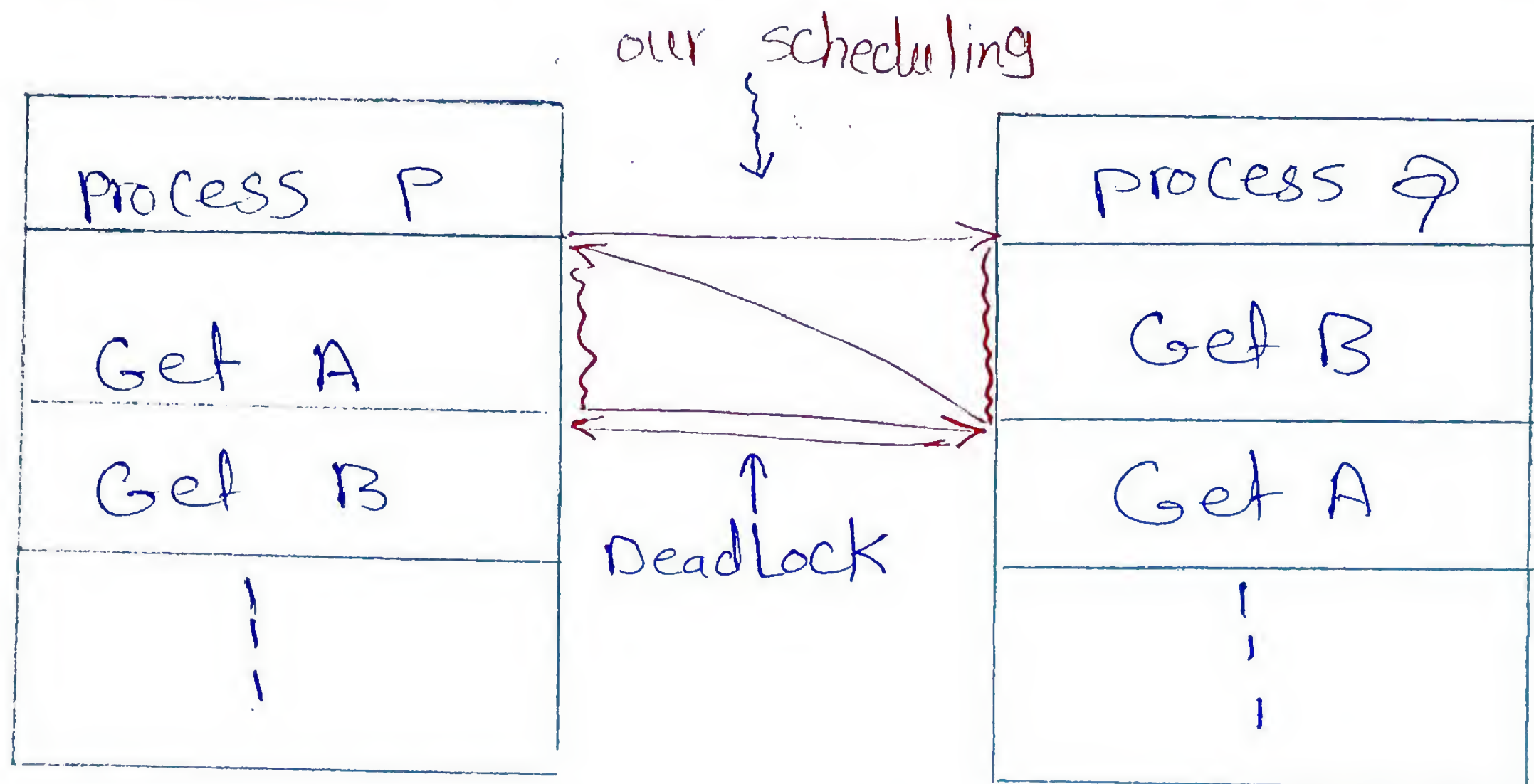
handling Mutual exclusion (concurrency control) between transactions:



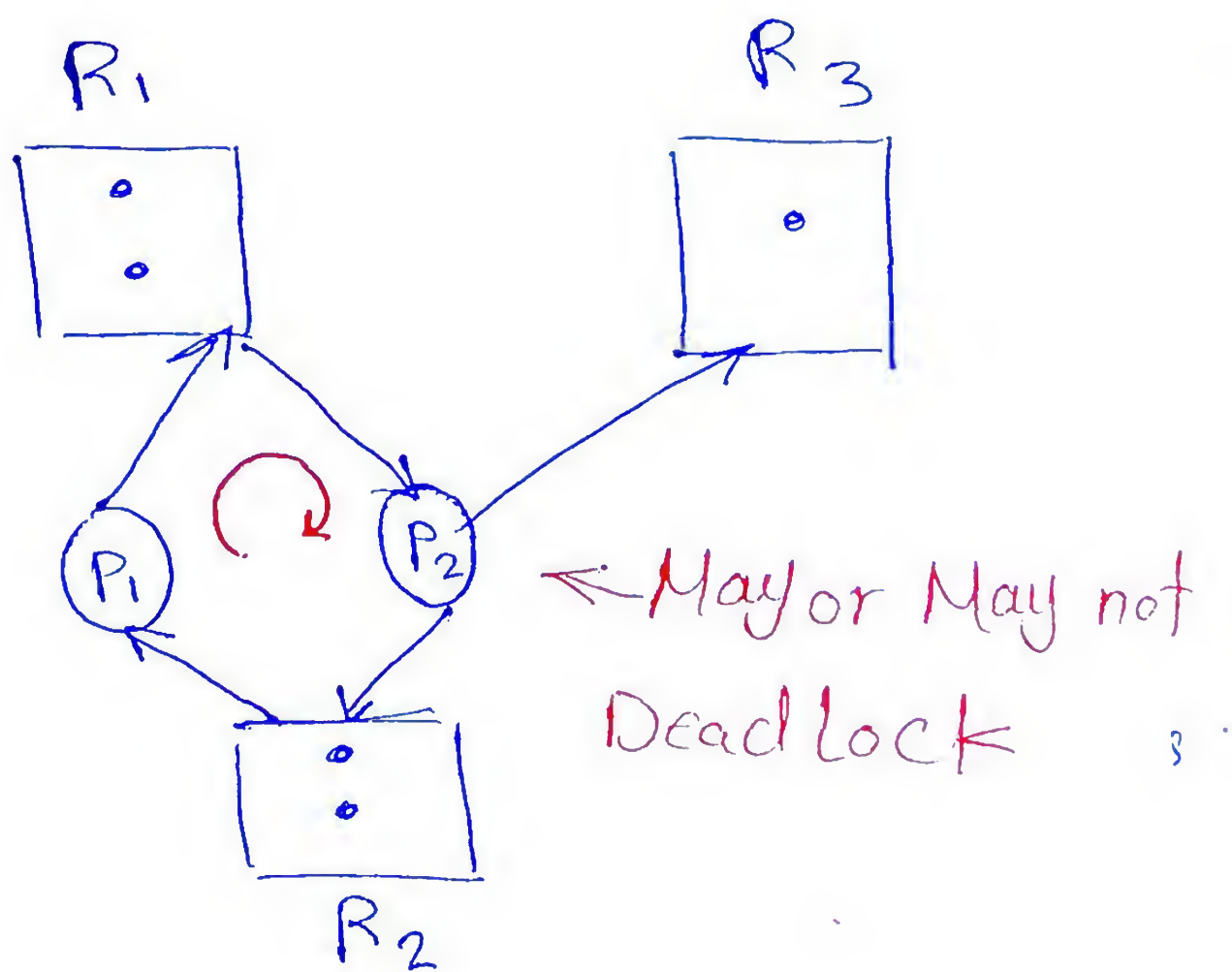
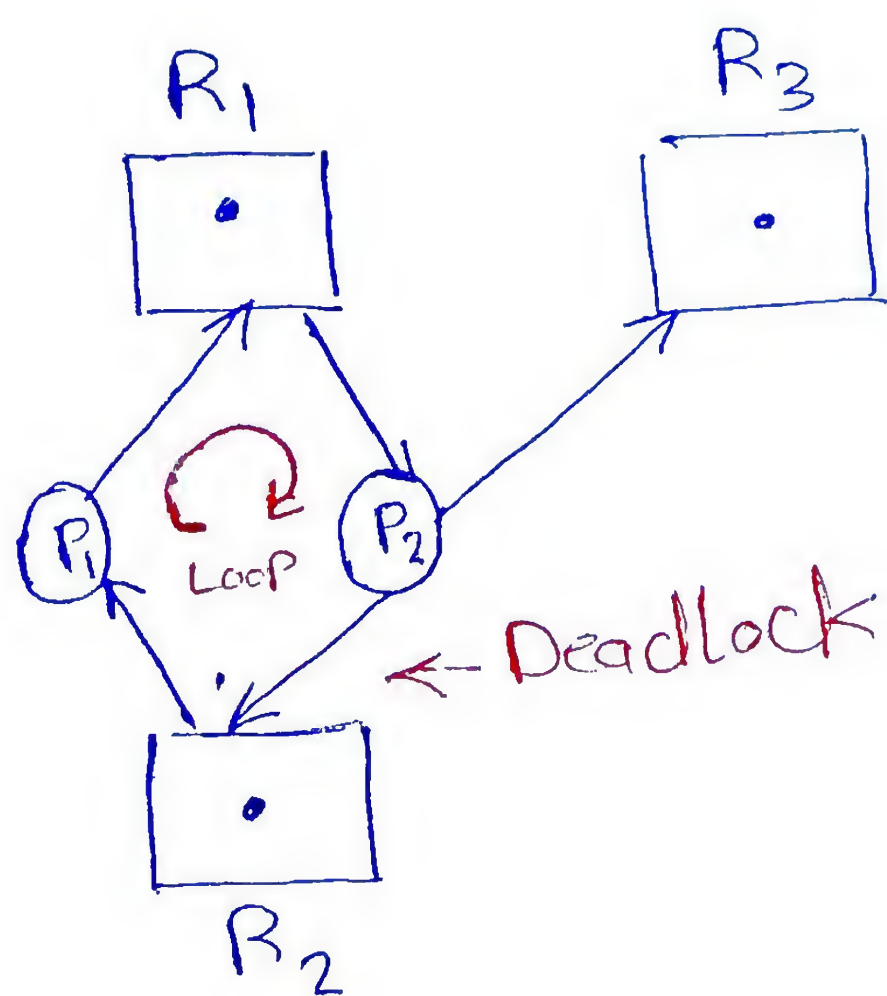
Three Methods of Scheduling (Concurrency Control)

Lock / Release	optimistic	Time Stamp
<p>When a process access any resource, it locks it and till it finishes the task, then releases it, so I ensure from Mutual exclusion.</p>	<p>every process ^{assume} no others works on the resources and if the resources status changed during transaction by others, the transaction is aborted, otherwise it's committed.</p>	<p>every process give the accessed resources its time stamp when accessing it, and if the resource has higher timestamp at any moment, meaning another one access it so it's aborted, otherwise it's committed.</p>

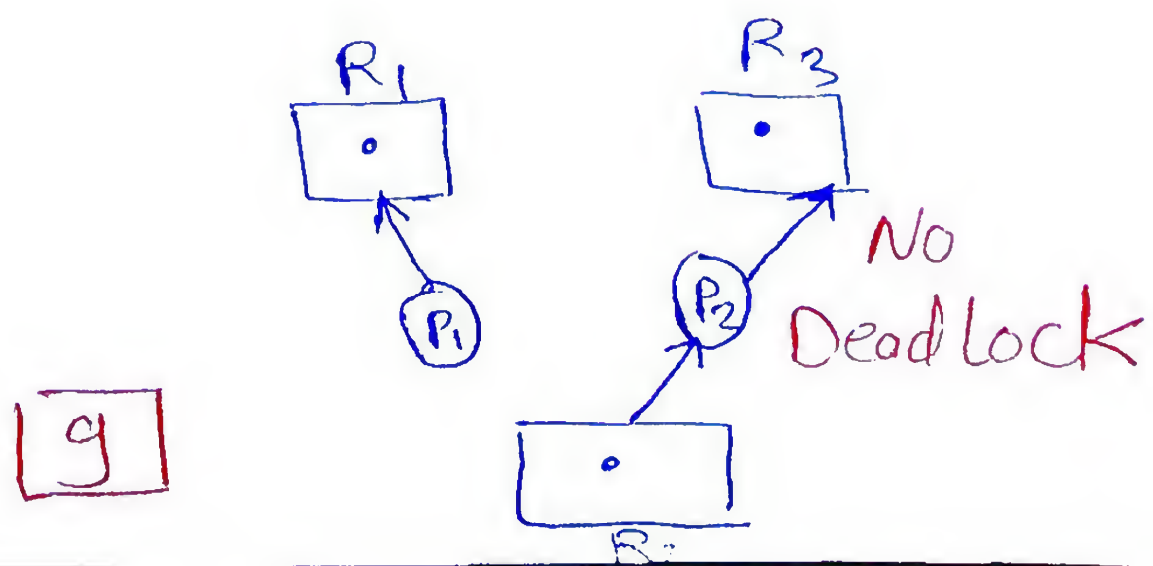
the Deadlock problem: a process hold the Resource **A** and wait for the resource **B** which is hold by another process waiting Resource **A**. :-



Resource Allocation Graph



□ : Mean Resource
 ○ : Process
 • : n. of Replication
 ○ → □ : Request resource
 □ → ○ : Resource held by this process



Dead Lock Characterization

« not understanding yet »

U

slide number (83)